Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Absolvování individuální praxe

Individual professional practice in the company

2014

Petr Juřica

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Juřica**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: VÍTKOVICE IT SOLUTIONS a.s.
2. Struktura závěrečné zprávy:
a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c) Zvolený postup řešení zadaných úkolů
d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vedl odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. RNDr. Petr Šaloun, Ph.D.**

Konzultant bakalářské práce: Ing. Milan Juřík

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *4. května 2014*

.....
podpis studenta

Poděkování

Rád bych poděkoval Ing. Milanu Juříkovi za umožnění absolvování praxe. Rád bych také poděkoval doc. RNDr. Petru Šalounovi, Ph.D. a týmu Ing. Marka Gáby za odbornou pomoc, konzultaci a cenné rady při tvorbě této bakalářské práce.

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této bakalářské/diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava."

Dne: *4. května 2014*

.....
podpis zástupce

VÍTKOVICE
VÍTKOVICE IT SOLUTIONS a.s.
Cihelní 1575/14
702 00 Ostrava-Moravská Ostrava
IČ 28606582
DIČ CZ28606582

Abstrakt

Tato práce popisuje průběh mé odborné praxe ve firmě Vítkovice IT Solutions. Bakalářskou odbornou praxi jsem vykonával od 19. září 2013 do 18. dubna 2014. V první části se zaměřuji na informace o firmě, její působení a zaměření. Dále uvádím všechny úkoly, na kterých jsem se po dobu vykonávání praxe podílel a jejich řešení. Ke konci této práce jsem popsal znalosti získané při studiu na VŠB, které jsem uplatnil při praxi, zkušenosti nově nabyté a také hodnotím přínos celé praxe.

Klíčová slova

C#; WPF; XAML; Odborná praxe; Vítkovice IT Solutions; Databinding; Stromová navigace; Výběrové pole; Zaškrtávací pole; Přehledová mapa

Abstract

This thesis describes the process of my individual professional practise in the company Vítkovice IT Solutions. I performed bachelor professional practice from 19th September 2013 to 18th April 2014. In the first part I'm focusing on information about company, its operations and focus. Furthermore I mention all the tasks that I've been involved and solutions for them. By the end of this paper, I described the knowledge that I gained while studying at the VŠB, and which I applied in practise and knowledge acquired in practise. At the end I also evaluate the contribution of the whole professional practise.

Key words

C#; WPF; XAML; Individual professional practice; Vítkovice IT Solutions; Databinding; Tree View; ComboBox; Checkbox; Overview map

Seznam použitých zkratek

Zkratka	Význam
C#	Programovací jazyk od firmy Microsoft
WPF	Windows Presentation Foundation. Technologie pro tvorbu grafických komponent
XML	Extensible Markup Language
XAML	Extensible Application Markup Language
JSON	JavaScript Object Notation

Obsah

Úvod

Běžnou praxí firem je požadovat po uchazečích několikaletou praxi. Z tohoto důvodu jsem se rozhodl získat odbornou praxi již při studiu a její absolvování jsem si zvolil ve firmě Vítkovice IT Solutions. Při výběru této firmy jsem využil školní informační systém NETFEI. Hlavním důvodem výběru této firmy byla její všeobecná proslulost a příslib nových znalostí v jazyce C#. Na pohovoru mi byla dána možnost volby, zdali chci pracovat na samostatné úloze, anebo pracovat v jednom z firemních týmů. S ohledem na možnost získání nových zkušeností od členů týmu, jsem zvolil druhou možnost.

Po úspěšném absolvování vstupního pohovoru a povinného školení o bezpečnosti práce jsem 19. září 2013 zahájil svůj první den praxe. Byl jsem zařazen do týmu Ing. Marka Gáby, jenž se zaměřuje na vývoj geografických informačních systémů za použití systému ArcGis od firmy ESRI. Tým se skládá ze dvou programátorů, jednoho odborníka na databáze a projektového manažera.

Zadané úkoly jsem prováděl samostatně a přímo na pracovišti. Výsledky své práce jsem předával svému nadřízenému programátorovi, panu Ing. Jaromíru Musiolovi. Díky znalostem programování v jazyce C#, které jsem získal při studiu na VŠB, jsem byl schopen se zapojit jako programátor do vývoje komerční aplikace, který probíhal právě v tomto jazyce.

1.1 O firmě

Společnost Vítkovice IT Solutions je ostravská firma zabývající se vývojem informačních a komunikačních technologii se sídlem v Ostravě. Vznikla spojením firem Vítkovice IT Solutions, Medium Soft a Netpropys.

Primárním odběratelem služeb jsou společnosti patřící do skupiny Vítkovice Machinery Group. Skupina se od svého vzniku zaměřila na co nejefektivnější vytvoření společné komunikace a procesů s důrazem na kvalitu poskytovaných služeb a vzájemnou efektivní komunikaci se zákazníkem. Společnost v průběhu krátké doby získala pro kvalitu svých nabízených služeb potřebné ISO certifikáty a osvědčení NBÚ.[1]

2 Překlad programu Geocortex

2.1 Latitude Geographics

V době mého nástupu na praxi firma jednala s kanadskou společností Latitude Geographics o zakoupení licencí k jejich programu Geocortex a možnost být výhradním distributorem softwaru Geocortex pro Českou republiku a Slovensko s možností prodeje do Polska a Maďarska.[2]

2.2 Přednáška Johna Fletchera

Kanadská firma poslala svého pracovníka Johna Fletchera, aby nám předvedl možnosti jejich programu a způsobu práce s ním. Přestože jsem byl u firmy pouze druhým dnem, bylo mi umožněno zúčastnit se prezentace, která probíhala v angličtině. Přes veškerou snahu pana Fletchera zjednodušit používaná slova, musel mnohdy volit složitější technické výrazy, kterým moji kolegové ne vždy rozuměli. Díky znalostem technické angličtiny, získaných při studiu na Vysoké škole Báňské, jsem byl schopen překládat složitější části prezentace a také zpětně tlumočit veškeré dotazy kolegů. Mého vedoucího velmi zajímala možnost přeložení celého programu. Velmi zajímavá prezentace trvala tři dny a mimo jiné obsahovala ukázky programu pro export všech řetězců, které bylo třeba přeložit, a také způsob jakým vytvořit instalátor klienta s vložením námi vytvořeného překladu. Ten byl pro nás velmi důležitý, neboť bez překladu by bylo velmi těžké software nabízet českým firmám a úřadům.

2.3 Překlad programu

Firma nám povolila vytvořit vlastní překlad a dokonce nás požádala o zaslání hotové verze, kterou by rádi od další verze nativně podporovali. Bylo třeba přeložit tři verze klientů a to jmenovitě: HTML5, Silverlight a Flex. Veškerý text byl uložen v souboru typu JSON. Každá verze měla svůj vlastní instalátor, do něhož bylo potřeba přidat nové jazykové soubory a do konfiguračního souboru přidat cestu k novému jazyku a opatřit ho správným jazykovým kódem. Zde nastal problém při zkoušení funkčnosti češtiny na různých zařízeních a prohlížečích. Čeština nefungovala na jednom ze třech nejčastěji používaných prohlížečů a na mobilech s operačním systémem Android. Problém jsem vyřešil přidáním jak dvoumístného, tak čtyřmístného jazykového kódu.

Při překladu jsem přeložil více než 900 řádků. Překlad i s korekcemi mně trval přibližně 10 dnů. U řetězců nebyl uveden jejich kontext a místo použití, tudíž jsem musel mnohdy odhadovat význam vět.

3 Zadané pracovní úkoly

Po dobu mé praxe mně bylo zadáno množství menších úkolů, ve kterých jsem vytvářel grafické komponenty do integrovaného záchranného systému vyvíjeného pro Zlínský kraj. Vzhled pro všechny komponenty byl vytvořen ve WPF za pomoci značkovacího jazyka XAML a jejich funkčnost byla zajištěna jazykem C#.

3.1 Tvorba stromové navigace a jeho obsahu

Tvorba stromové navigace byla mým prvním úkolem. Tvoří jednu z nejdůležitějších součástí programu. Stromová navigace byla využita na dvou místech ve finální aplikaci. Do prvního stromu se vkládají aktivní události, neboli místa kde se stala nehoda a kde je potřeba poslat výjezdovou skupinu. Ve druhém stromě jsou výše zmíněné výjezdové skupiny.

3.2 Tvorba filtrovatelného Výběrového pole

Dalším důležitým komponentem aplikace bylo výběrové pole, ve kterém se filtrovaly prvky podle zadaného textu. Na návrh analytika byla přidána možnost výběru, zdali se má filtrovat podle prefixu slova, anebo jestli zadaný text je podslovem některého z prvků.

3.3 Mapa posouvatelná myší

Pro zlepšení práce s aplikací mně byl zadán úkol vytvořit náhlednou mapu, která by byla posouvatelná myší. Řešením bylo přidání interaktivního proužku nad mapu.

Zadaný úkol	Počet člověkodnů
Překlad programu Geocortex	10
Stromová navigace	21
Filtrované výběrové pole	13
Posouvatelná mapa	5
Školení a prezentace	5

Tabulka 1.1: *Náročnost jednotlivých úkolů*

4 Řešení zadaných úkolů

4.1 Stromová navigace

Mým prvním úkolem bylo vytvoření tzv. stromové navigace, do které se budou vkládat události a výjezdové skupiny. Tento úkol byl zároveň mým úvodem do tvorby grafických prvků ve WPF. Při studiu jsem se v předmětu Programovací jazyky II. setkal s technologií, která WPF předcházela, a to WinForms. WPF oproti WinForms nabízí daleko větší možnosti v úpravě vzhledu komponenty. Kód je rozdělen na část, v níž se popisuje vzhled stromu a na kód, který běží na pozadí a určuje celkovou funkčnost stromu.

Základem bylo vytvořit si nový soubor typu Control, což je samostatný interaktivní prvek s grafickým rozhraním, který se dá lehce vkládat na jakékoliv místo v aplikaci. UserControl je typ kontroly, která dědí většinu metod, vlastností a událostí ze svého předka Control.[3] Stromová navigace je již v základní podobě předem implementována a rozhodl jsem se ji použít, bylo však potřebné změnit vzhled jednotlivých prvků stromu.

4.1.1 KorenViewModel

Základem každého stromu je kořen. V projektu jej zastupuje třída KorenViewModel

```
3 references
public class KorenViewModel
{
    1 reference
    public List<ObjektViewModel> KolekceKorenu { get; private set; }
    2 references
    public KorenViewModel(Objekt koren)
    {
        var korenStromu = new ObjektViewModel(koren);
        KolekceKorenu = new List<ObjektViewModel> {korenStromu};
    }
}
```

Obrázek 1.1: *Ukázka zdrojového kódu třídy KorenViewModel*

Stromová navigace při inicializaci vyžaduje kolekci prvků, kterou jí předávám pomocí vlastnosti KolekceKorenu. Konstruktoru této třídy se předává instance třídy Objekt, která je již naplněna odkazy na své potomky. Vlastnost KolekceKorenu je poté volána při inicializaci základního prvku stromové navigace.

4.1.2 Objekt

Tato třída představuje každý prvek stromové navigace a popisuje jejich vlastnosti a obsahuje kolekci všech potomků.

4.1.3 ObjektViewModel

ObjektViewModel je prostředníkem mezi XAML soubory a třídou Objekt a poskytuje rozhraní, pomocí kterého nám umožnuje za pomoci kódu měnit obsah a vzhled prvků stromové navigace. Třída obsahuje všechny vlastnosti, které si přejeme měnit za běhu programu. Moje prvotní pokusy o změnu barvy pozadí za běhu aplikace dopadly neúspěšně. Aplikace na změny nereagovala. Při hledání chyby jsem za pomoci krokování procházel jednotlivé fáze běhu aplikace a zjistil jsem, že dané vlastnosti byly změněny, avšak grafická komponenta tyto změny nezaznamenala. Řešením bylo implementování rozhraní INotifyPropertyChanged.

```
public event PropertyChangedEventHandler PropertyChanged;

4 references
private void PriZmneneVlastnosti(String nazevVlastnosti)
{
    if (PropertyChanged != null)
        PropertyChanged(this, new PropertyChangedEventArgs(nazevVlastnosti));
}
```

Obrázek 1.2: *Ukázka implementace INotifyPropertyChanged*

Pro korektní implementaci je potřeba dvou věcí. Událost PropertyChanged a metoda, která s výše zmíněnou událostí pracuje. Metoda se spustí při změně vlastnosti a upozorní klienta na změnu obsahu nebo vzhledu grafické komponenty. Slabinou této metody je nutnost předávat název vlastnosti za pomoci řetězce a v případě změny názvu vlastnosti nebo překlepnutí kompilátor není schopen rozpoznat chybu. Firma Microsoft proto ve verzi 5 jazyka C# přidala atribut CallerMemberName, který automaticky zjistí, která vlastnost zavolala metodu a předá její název[4].

```
public event PropertyChangedEventHandler PropertyChanged;

4 references
private void PriZmneneVlastnosti([CallerMemberName] string nazevVlastnosti = "")
{
    if (PropertyChanged != null)
        PropertyChanged(this, new PropertyChangedEventArgs(nazevVlastnosti));
}
```

Obrázek 1.3: *Ukázka implementace INotifyPropertyChanged ve verzi C# 5.0*

4.1.4 UdalostiControl

UdalostiControl je stromová navigace pro události. Třída se skládá z části, která popisuje vzhled komponenty a části, která se stará o funkčnost celé komponenty.

Soubor UdalostiControl.xaml se celý skládá z grafického popisu stromové navigace a úpravy vzhledu jednotlivých uzlů stromu. Na začátku jsem nastavil parametry prvku TreeView. V tomto místě jsem se poprvé setkal s DataBinding, které slouží k provázání dvou vlastností. Jakmile se změní hodnota jedné vlastnosti, změní se i všechny provázané vlastnosti [5]. U ItemsSource jsem jako vstupní parametr použil vlastnost KolekceKorenu ze třídy

KorenViewModel. Abych mohl se stromem dále pracovat na jiných místech v programu, bylo nutno mu zadat název. V rámci zjednodušení práce s aplikací jsem nastavil, aby byl strom při inicializaci automaticky rozbalený.

Ke každému objektu jsem implementoval zaškrtávací pole, obrázek, expander a slovník uvnitř expanderu (viz. obrázek 1.4). Rozhodl jsem se využít prvku HierarchicalDataTemplate, který popisuje visuální strukturu víceúrovňových komponent[6]. Pomocí něj jsem nastavil každému prvku šablonu, podle níž bude vytvořen. Nejdříve jsem nastavil horizontální řazení prvků za pomoci atributu Orientation v rozložení StackPanel, jenž funguje jako zásobník pro grafické komponenty.

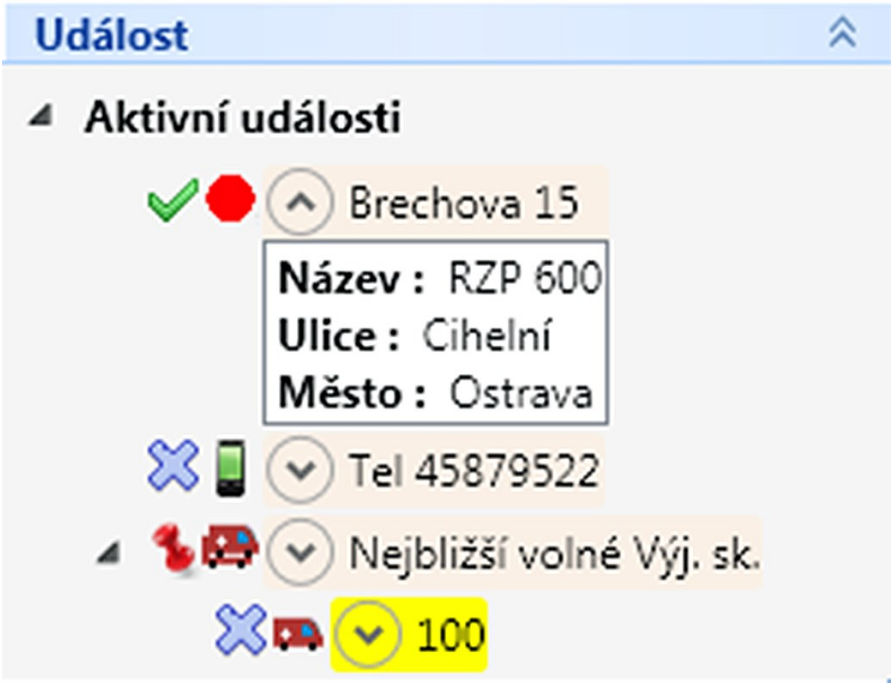
Následovalo vytvoření zaškrtávacího pole. Jedním z nejdůležitějších požadavků u této komponenty byl tzv. třetí stav. WPF tento třetí stav nativně podporuje a je možné jej povolit pomocí nastavení hodnoty True u atributu IsThreeState. Pro každý stav jsem vytvořil událost, která do ladící konzole vypíše ID zaškrtnutého objektu a uvede stav zaškrtávací pole. Při vkládání prvků do stromu je potřeba některá zaškrtávací pole vytvořit s předem zadanými stavy, které se mohou lišit od defaultního. Nastavení se provádí za pomoci atributu IsChecked, který jsem pomocí DataBindingu provázal s vlastností třídy Objekt. Povolením třetího stavu se změnila defaultní hodnota zaškrtávacího pole, protože třetí stav se nastaví vložením hodnoty Null a tím, že stav zaškrtávacího pole nastavujeme pomocí vlastnosti Zaskrtnuto typu Boolean, která může nabývat hodnoty Null. Pokud tedy při vkládání prvků do stromu nenastavíme zaškrtávacímu poli hodnotu, tak zobrazí třetí stav, což není korektní. Problém jsem vyřešil nastavením defaultní hodnoty vlastnosti Zaskrtnuto na False v konstruktoru třídy Objekt. Dalším úkolem bylo změnit defaultní ikony, které se zobrazují pro jednotlivé stavy zaškrtávacího pole. WPF úpravu grafických komponent podporuje a umožňuje u komponent změnit téměř cokoliv. Pro přístup k ikonám zaškrtávacího pole jsem použil prvek CheckBox.Style a s pomocí prvku Setter jsem nastavil ikonu defaultnímu stavu. Poté jsem vytvořil triggery, které podle stavu, v němž se zaškrtávací pole nachází, zobrazí příslušnou ikonu.

Další částí uzlu stromu byl obrázek, který má symbolizovat typ události. Obrázek je implementován pomocí prvku Image. Nastavil jsem u něj maximální výšku a šířku na 16 pixelů, aby nepřesáhl velikostí ostatní prvky uzlu. Je načítán pomocí URI adresy předané ze svázané vlastnosti ImageUri.

Posledním prvek je expander. Zabalený expander se skládá pouze z nadpisu a tlačítka, po jehož stisknutí se podle zadaného směru celý rozšíří a zobrazí svůj obsah. Uvnitř jsou informace o události. Informace jsem ukládal ve formě slovníku se dvěma řetězci. Pro lepší přehlednost jsem u prvního řetězce nastavil tučné písmo. V prvotních verzích jsem tučné písmo řešil za pomoci objektu Run, který se používá pro náročnější úpravy a formátování textu. V mém případě však byla implementace objektu Run zbytečná a pouze zpomalovala aplikaci, samotný TextBlock, který se používá pro výpis textu, bohatě stačil.

Dalším prvkem Stromové navigace je nadpis využívaný pouze v kořenu stromu. U kořene stromu není nutno použít např. expander nebo zaškrtávací pole. S mými znalostmi jsem

nenalezl způsob, jak upravit šablonu, podle které se vyrábí všechny prvky, aby učinila výjimku při kořenu stromu a nepřiřadila mu výše zmíněné prvky. Jediné řešení, které mně napadlo, bylo schování daných prvků, kvůli čemuž jsem musel vytvořit další dvě vlastnosti, nastavující jejich viditelnost. Při schování expanderu však také ztrácíme text u prvku, který byl dodáván nadpisem expanderu. Tudíž jsem vytvořil nový TextBlock, jenž se zobrazí pouze v případě, pokud hodnota vlastnosti Koren není prázdná. Elegantnější řešení tohoto problému je předmětem dalšího studia.



Obrázek 1.4: *Ukázka naplněné stromové navigace*

4.2 Filtrované výběrové pole

Po dokončení stromové navigace mně byl zadán úkol vytvořit filtrované výběrové pole. Výběrové pole má v sobě zabudovanou vlastní implementaci filtrování podle zadaného textu, avšak velmi omezenou. Filtrování funguje správně, ale nezobrazuje okno s filtrovanými prvky, a pokud dva prvky mají například stejnou předponu, tak výběrové pole automaticky zvolí první prvek. Tato omezení nás vedla k tvorbě vlastního výběrového pole. Jazyk C# umožňuje upravovat předem implementované základní komponenty jako výběrové pole, obrázek aj.

V projektu jsem založil třídu FiltrovanyComboBox, a aby bylo možné třídu použít jako výběrové pole, musí dědit ze třídy ComboBox. Poté je možné takto vytvořenou komponentu vytvořit v aplikaci.

```
<local:FiltrovanyComboBox
 x:Name="CmbAuto" IsEditable="True"
 ItemsPanel="{DynamicResource ItemsTemplate}" Height="23"
 Margin="19,112,0,0"  VerticalAlignment="Top" HorizontalAlignment="Left" Width="91" />
```

Obrázek 1.5: *Ukázka implementace filtrovaného výběrového pole*

První metoda, jež se po spuštění aplikace provede, je metoda OnItemsSourceChange, která se spustí vždy, když změníme obsah výběrového pole. Metoda přiřadí novému obsahu filtr v podobě metody PredikatFiltru. Celé přiřazení je v podmínce, která kontroluje, zda je pole naplněno, protože metoda se poprvé spustí při vytvoření komponenty, když ještě není naplněna žádnými prvky a aplikace se pokoušela zbytečně filtrovat prázdné výběrové pole. Další ze zděděných metod je OnApplyTemplate, která se spouští v momentě, kdy aplikace tvoří komponentu. V této metodě jsem navázal událost k výběrovému poli, která zavolá metodu ObnovitFiltr pokaždé, když se změní zadaný text. Dále jsem navázal událost puštění klávesy na metodu PusteniKlavesy.

Metoda ObnovitFiltr se spustí při změně zadaného textu a znovu provede filtraci dle nového textu. Filtr nejprve zjistí, jaký typ prvku kontroluje. V zaškrtávacím poli mohou být jak číselné hodnoty, tak objekty typu PrvekRuian. Po zjištění typu prvku, aplikace zavolá metodu, která zjistí, zdali má vyhledávat podle prefixu slova či podslova. Pokud testovaný prvek splňuje podmínku, zůstává dále v kolekci prvků.

Nativní filtrování bylo zaměřeno především na filtrování dle prvního zadaného znaku. I když jsem v metodě OnApplyTemplate zakázal dané vyhledávání, část jeho funkčnosti stále běžela na pozadí. Projevovalo se to automatickým označením prvního zadaného znaku. Tento jev je velmi rušivý, pokud chceme vyhledávat podle dvou a více znaků. Z tohoto důvodu jsem implementoval metodu PusteniKlavesy, která v případě označení prvního znaku automaticky označení zruší a dovolí uživateli psát další znaky bez nechtěného smazání prvního.

Při práci na tomto úkolu jsem se poprvé setkal s DependenyProperty, což je vlastnost, která v sobě obsahuje implementovaný návrhový vzor Observer. DependencyProperty jsem použil při tvorbě vlastnosti, jenž v sobě ukládá informaci, jestli je DropDown okno otevřeno. Tato vlastnost je využita v metodě OnGotFocus, v níž se podmínkou dotážu, zda výběrové pole obsahuje prvky a DropDown, okno není otevřeno a pokud je podmínka splněna, pak metoda otevře DropDown okno. Tím zajistím, že po kliknutí na výběrové pole se otevře okno s výpisem všech prvků pole.

4.3 Posouvatelná Mapa

Hlavním bodem aplikace je mapa, na které se zobrazují všechny události a výjezdové skupiny. Pro lepší práci s touto mapou je v pravém horním rohu aplikace implementována přehledová mapa. Mým úkolem bylo umožnit přesouvání přehledové mapou, aby si jí mohl uživatel přesunout na pro něj vyhovující místo. Můj vedoucí mi napověděl část správného řešení, a to použití prvku Border.

Mým plánem bylo umístit nad přehledovou mapu interaktivní proužek, který by byl responzivní vůči stisknutí a pohybu myší a celý tento pohyb by přenášel na nadřazený panel, v němž je spolu s přehledovou mapou umístěn. Při práci na tomto úkolu jsem využil znalosti různých typů grafických rozhraní a jejich skládání z předmětu Uživatelské rozhraní.

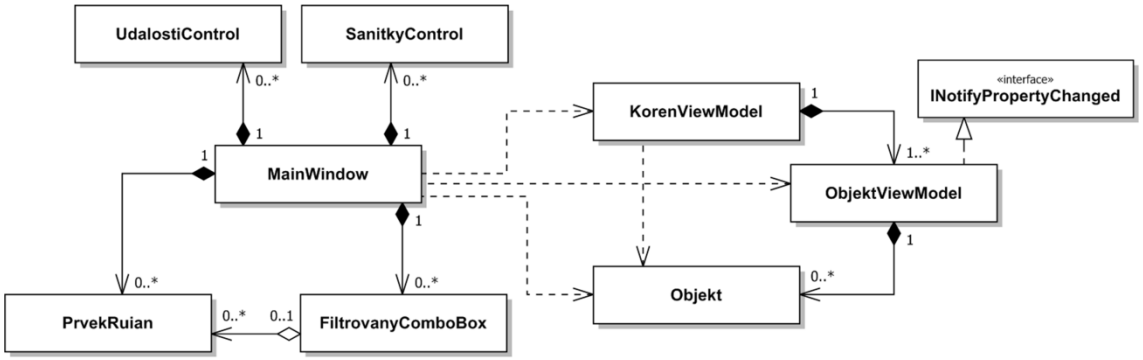
Jako první jsem vytvořil komponentu DockPanel, která mi usnadnila práci s umístěním přehledové mapy a interaktivního proužku. Pohyb s DockPanelem jsem vyřešil pomocí vlastnosti RenderTransform, jež popisuje, jak se má prvek vykreslovat na obrazovce. Dále DockPanel obsahuje Grid pro interaktivní proužek a další Grid pro přehledovou mapu. Grid sám o sobě se v aplikaci nevykresluje a pouze ohraničuje do něj vložené prvky, a proto jsem do prvního Gridu vložil, mým vedoucím navrhnutý, objekt Border, který vyplňuje celý Grid. Na objekt Border jsem navázal události pro stisknutí levého tlačítka myši, pohyb myší a puštění levého tlačítka myši.

Při stisknutí levého tlačítka program přichytí myš k proužku a do proměnné se uloží počáteční pozice kurzoru myši a DockPanelu. Při posunu myší odečítám její novou pozici od počáteční a tento rozdíl připočítávám k pozici DockPanelu.

Při vytváření komponenty DockPanel se také kromě RenderTransform nabízelo využití LayoutTransform. Hlavním rozdílem mezi těmito dvěma vlastnostmi je pořadí provedení akcí. Přesun pomocí LayoutTransform ovlivňuje okolní prvky. Každý pohyb nejdříve přearanžuje okolní prvky a až poté umístí námi přesunutý prvek, což zapříčiní zvýšení nároků aplikace. RenderTransform naopak dovolí prvku překrývat ostatní prvky a neovlivňuje jejich pozici. Z důvodu snížení nároků jsem se rozhodl pro RenderTransform. Jediným nedostatkem tohoto řešení je fakt, že přehledová mapa nyní může být přesunuta mimo hlavní mapu. Kvůli nedostatku času jsem nestihl tuto chybu opravit.

4.4 Popis testovací aplikace

Po celou dobu praxe jsem vyvíjel v testovací aplikaci, kterou přikládám k této práci. Z důvodu licenčních problémů v této aplikaci nejsou umístěny mapy a tak do místa, kde měla být mapa, jsem umístil čtyři tlačítka, která mně sloužila k testování funkčnosti aplikace. První tlačítko naplní stromovou navigaci testovacími hodnotami, druhé tlačítko se pokusí tyto hodnoty přepsat, třetí tlačítko uloží stavy všech zaškrtávacích polí a poslední tlačítko nastaví u všech zaškrtávacích polí námi uložené stavy. Pod tlačítky jsou dále dvě výběrová pole, každé naplněno různými typy proměnných a pod nimi je zaškrtávací pole, které mění způsob, jakým se filtrují prvky. Na pravé straně aplikace je okno s přehledovou mapou. V době ukončení mé praxe obsahovala má testovací aplikace 8 tříd.



Obrázek 1.6: *Třídní diagram aplikace*

Třídy	Index Udržitelnosti	Cyklomatická složitost	Provázanost	Počet řádků kódu
MainWindow	56	37	52	110
FiltrovanyComboBox	75	25	31	43
PrvekRuian	94	5	0	5
KorenViewModel	90	3	3	5
Objekt	92	27	5	29
ObjektViewModel	89	33	14	48
UdalostiControl	82	4	8	9
SanitkyControl	83	1	3	3
Celkově	83	116	88	252

Tabulka 1.2: *Výpis složitosti kódu*

Hodnoty v tabulce 1.2 byly naměřeny za pomoci programu Code Metrics Viewer, který je součástí Visual Studia.

5 Chybějící znalosti

Při nástupu do firmy jsem postrádal znalosti o Geoinformačních systémech, které se sice na Vysoké škole Báňské vyučují, ale tento předmět jsem si bohužel ve druhém ročníku nezapsal. Kvůli tomu jsem se nepodílel na vývoji celé aplikace, ale vyvíjel jsem grafické komponenty. WPF a XAML jsem poznal teprve na praxi a nejtěžší částí na naučení pro mne byla dynamická typová kontrola v jazyce XAML, Databinding.

6 Znalosti a dovednosti získané během studia a uplatněné v průběhu odborné praxe

V průběhu praxe jsem při tlumočení přednášky Johna Fletchera a programu Geocortex využil znalosti technické angličtiny získané při studiu. Znalosti Objektově orientovaného programování, které jsem získal v předmětech Algoritmy II. a Vývoj informačních systémů mně také velmi pomohly při tvorbě stromové navigace. Nejdůležitější znalosti jsem získal z předmětu Programovací jazyky II., kde jsem se naučil základy jazyka C# a XML. V rámci studia rovněž absolvoval předmět Uživatelské rozhraní, který mi pomohl s grafickým rozložením prvků a použití vhodného layoutu. V předmětu Programovací jazyky I. jsem se naučil, jak se správně zorientovat v cizím kódu. Při obhajobách různých projektů jsem se naučil prezentovat svou práci, vysvětlit funkčnost programu a způsob jakým jsem dosáhl cíle.

Závěr

Všechny zadané úkoly jsem splnil až na poslední úkol s pohybovou mapou, kdy už bohužel nezbyl čas na dořešení posledních problémů. Vzhledem k tomu, že mně byla nabídnuta možnost další spolupráce s firmou Vítkovice IT Solutions, otevírá se tím příležitost k seznámení se s alespoň již fungujícím způsobem jejich vyřešení.

Za možnost absolvování odborné praxe jsem velmi vděčný, neboť mi umožnila nahlédnout do fungování proslulé firmy, získal jsem neocenitelné zkušenosti při práci na komerčním programu, umožnila mi prohloubit znalosti a navázal jsem nové pracovní vztahy. Velkým přínosem mi také byla zkušenost se zapojením do již běžícího projektu. Celkově hodnotím praxi jako velmi přínosnou a obohacující. Jedním z největších úskalí bylo skloubení docházky na praxi se studijními povinnostmi a tím pádem nutnost správně si zorganizovat čas a rozvrh.

Použitá literatura

[1] VÍTKOVICE IT SOLUTIONS A. S. Vítkovice IT Solutions [online]. © 2011 [cit. 2014-04-30]. Dostupné z: http://itsolutions.vitkovice.cz/

[2] GIS: Geografický Informační Systém. Vítkovice IT Solutions [online]. 2013 [cit. 2014-04-30]. Dostupné z: http://itsolutions.vitkovice.cz/37/cs/node/3401

[3] ŠTURALA, Aleš. 6. WPF: vytváříme kontroly. Vyvojar.cz [online]. [cit. 2014-04-30]. Dostupné z: http://www.vyvojar.cz/Articles/460-6-wpf-vytvarime-kontroly.aspx

[4] HRNJICA, Bahrudin. New feature in C# 5.0: [CallerMemberName]. In: Bahrudin Hrnjica Blog[online]. 18. 03. 2012 [cit.2014-04-30]. Dostupné z: http://bhrnjica.net/2012/03/18/new-feature-in-c-5-0-callermembername/

[5] Lekce 7: DataBinding a formuláře. Výukový kurz Microsoft Silverlight [online]. 2010 [cit. 2014-05-04]. Dostupné z: http://silverlight.cs.vsb.cz/07-databinding.aspx

[6] How to: Use a TreeView to Display Hierarchical Data. *MSDN : The Microsoft Developer Network* [online]. 2014 [cit. 2014-05-04]. Dostupné z: http://msdn.microsoft.com/en-us/library/dd759035(v=vs.95).aspx

Seznam příloh

Obsah přiloženého CD:

- Elektronická verze Bakalářské práce
- Zdrojový kód aplikace ve složce
- Třídní diagram aplikace ve formátu PNG. Soubor je uložen pod názvem DiagramTrid.png
- Ukázkové obrázky výsledné aplikace se zvýrazněnou částí, kterou jsem implementoval.
  Soubory jsou uloženy pod názvy: GIS ZZS - Udalost.png
  GIS ZZS – Vyhledavani v mistopisu.png
  GIS ZZS - Vyjezdove skupiny.png